Sevgili okuyucular,

Bu sayımızda sizlere, PostScript ve sayfa düzenleme dilleri hakkında bilgi vermeye çalıştık. Birbirleriyle sık sık karıştırılan bu iki kavram hakkında umarız sizleri biraz da olsa aydınlatabildik. Bu yazının hazırlanmasında emeği geçen ODTÜ Elektrik ve Elektronik Mühendisliği Bölümü öğrencilerinden Hakan Karakaş'a teşekkür ediyoruz.

Bu sayımızda ayrıca, Antalya Anadolu Lisesi öğrencilerinden klüp üyemiz Kürşat Aker'in fare kullanımı ile ilgili olarak hazırladığı programa yer veriyoruz.

Sizlerin de kısa program, yazı, duyuru, grafik, karikatür, fıkra ya da benzeri şeylerle klübe katkılarınızı bekliyoruz. Bunları gönderirken, daha önce klübümüze üye olanlar, üye numaralarını yazmayı unutmasınlar. Diğer okuyucularımız için üyelik çağrımız devam ediyor. Yazışma adresimiz:

Emrehan HALICI
Bilgisayar Klübü,
Bilim ve Teknik Dergisi,
Atatürk Bulvarı, No:221,
Kavaklıdere, Ankara

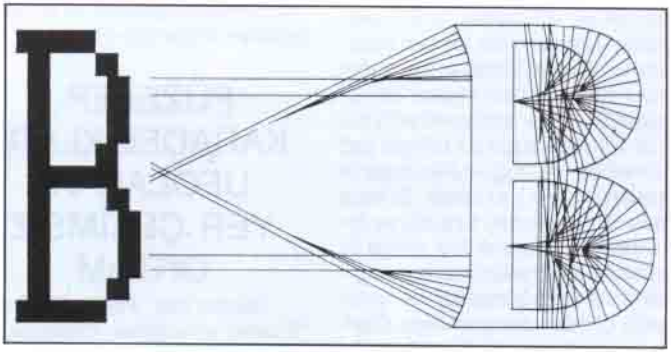
Her bir yazım biçiminin okuyanda etkisi farklıdır. Kullanıcılar, yazılarında değişik yazım biçimleri ve büyüklükler kullanmak isterler.

POSTSCRIPT NEDİR?

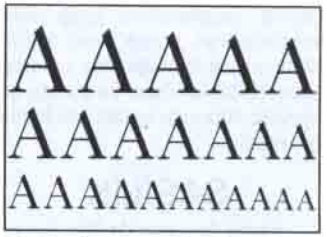
PostScript bugün en yaygın kullanılan sayfa düzenleme dilidir (Page Description Language). İlk kapsamlı sayfa düzenleme dili olması nedeniyle, Adobe firmasının PostScript'i üretirken kullandığı belirtimler, zamanla sayfa düzenleme dilleri arasında bir standart halini aldı. Tüm sayfa düzenleme dillerinin, sayfa üzerinde yer alacak noktaların kontrol edilmesi ilkesine göre çalıştığı göz önüne alınırsa, PostScript ile diğer sayfa düzenleme dilleri arasındaki farkın bir ilke farkından çok bir derece farkı olduğu görülür.

Sayfa üzerindeki her noktanın kontrol edilmesi, normal lazer yazıcılarda kullanılan nokta üretme yöntemlerinden oldukça farklıdır. Bir sayfa düzenleme diline sahip olmayan yazıcılarda, her bir harfin nasıl oluşturulacağını belirleyen bit eşleme (bitmap) tabloları bulunur. Bu bit eşlem tabloları sayesinde yazıcılar, herhangi bir karakteri oluşturan noktaların sayfanın neresine ve nasıl yerleştirileceğini bilirler. Ancak bu yöntemle döküm alabilmek için, kullanılan her karakter boyutu için ayrı ayrı bit-eşleme tablolarına ihtiyaç vardır. Bu tablolar, yazıcının standart belleğinde ya da yazıcıya takılıp çıkartılabilen font kartlarında saklanır. Bir başka seçenek olarak da bilgisayarda kullanılan yazılım programları tarafından yazıcıya gönderilir. Bu yollardan hangisi kullanılırsa kullanılsın, herhangi bir karakteri yazıcıdan bastırabilmek için o karakteri tanımlayan kısa emirlerin yazıcıya gönderilmesi gerekir. Bu emirleri alan yazıcı da sahip olduğu bit-eşlemelerini kullanarak o karakteri oluşturur.

Bu yöntemle nokta üretmek, birtakım problemleri de beraberinde getirir. Örneğin, her bit-eşleme tablosu, sadece bir karakter yazım biçiminin belli bir boyuttaki halini içerdiğinden, değişik boyut ve yazım biçimlerinde harfler kullanabilmek için, bu özellikleri sağlayan ayrı bit-eşleme tab-



Bit-eşleme ile oluşturulan bir karakter, aynı bit-eşleme tablosuna göre büyütülerek yazıldığında, biçiminde bozulmalar olur. Karakterler çizgi ve eğriler kullanılarak tanımlandığında, karakterin büyütülmesiyle biçiminde bir bozulma olmaz.



Sayfa düzenleme dilleri ile çok çeşitli büyüklüklerde harfler yazdırmak mümkündür.

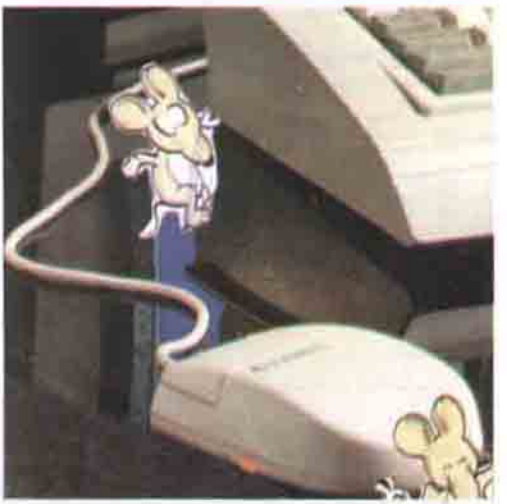
lolarına sahip olmak gerekir. Bu yüzden, yazılarımızda kullanabileceğimiz karakter büyüklüğündeki çeşitlilik hem yazıcının belleği, hem de o yazıcıyı destekleyen fontlardaki büyüklük çeşitliliği ile sınırlıdır. Değişik büyüklük ve biçimler dışında, arka planda fonlar oluşturmak gibi özel efektler kullanmak istiyorsak, bu efektleri tam olarak sağlayan bit-eşleme tabloları gerekir.

Sayfa düzenleme dilleri, bu tür değişik özellikler içeren karakterleri oluşturmak için bit-eşlemeden oldukça farklı bir yöntem kullanırlar. Her karakter ve onların değişik yazım biçimleri için ayrı ayrı bit-eşleme tabloları kullanırlar. Bu yazım biçimleri, karakter ve onların değişik yazım biçimleri için ayrı ayrı bit-eşleme tabloları kullanmak yerine, oluşturulacak karakterin ana hatlarını belirleyen temel yazım biçimlerini kullanırlar. Bu yazım biçimleri, karakterin kesin biçimini belirlemek yerine, o karakteri oluşturan temel öğelerin şekillerini ve yerlerini belirler. Örneğin, herhangi bir karakter için bu çizgi şuradan şuraya gidecek, oradan şu açıyla geri dönecek gibi temel bilgiler saklanır. Sayfa düzenleme dilleri bu temel yazım biçimlerini kullanarak, 2 nokta (1/36 inç) ile 200 nokta (3 inç) arasında değişen boyutlarda karakter oluşturabilir. Onları döndürebilir, eğebilir, gölgelendirebilir ya da arka kısımda değişik fonlar oluşturabilirler.

Sayfa düzenleme dilleri karakterleri tek tek yazıcıya göndermek yerine, tüm sayfayı bir seferde düzenleyerek yazıcıya gönderirler. Bu tür bir yöntem, tüm sayfanın düzenlendikten sonra bit-eşleme tablosunun hazırlanması ve tüm sayfanın tek bir karaktermiş gibi algılanması olarak düşünülebilir. Böyle bir bit-eşleme tablosunu alan yazıcı tüm sayfayı bir kerede oluşturur.

Tüm sayfanın tek bir seferde düzenlenip basılması, hızını olumsuz etkiler. Bilgisayarın tüm bir sayfayı yazıcının anlayacağı emirler haline getirmesi zaman alır. Sayfanın karmaşıklığı ve yoğunluğu bu zamanı ayrıca artırır. Eğer daha iyi bir döküm kalitesi için biraz zamandan fedakârlık etmeye razı iseniz, PostScript bir yazıcı sizin birçok isteğinizi yeterince tatmin edecektir.

PROGRAM: FARE KULLANIMI



Klüp üyelerimizden 0074-75-07 Kürşat Aker, Antalya Anadolu Lisesi'ne devam ediyor. 6 yıldır bilgisayar kullanıyor. Yapay zekâ konularına ilgi duyuyor. Aşağıda Kürşat Aker'in hazırladığı fare kullanımına ilişkin yazı ve programı veriyoruz. Eğer bir sorun çıkarsa veya daha fazla bilgi almak isterseniz ya da başka bir konuda yazışmak isterseniz Kürşat Aker'in adresi:

Kürşat AKER
3840. Sok. Yuva Apt. D:7
07050 ANTALYA

BİLGİSAYAR KLÜBÜ ÜYELERİ

Üye numaraları sıra no-doğum tarihi-il biçimindedir. İki adet resimleri eksik olanlar (r) ile gösterilmiştir. Bu üyelerimizden, en kısa zamanda arkasına isimlerini yazdıkları resimlerini bekliyoruz.

0101-76-34 Murat Büyük
0102-77-45 MuratYılmaz
0103-76-23 Nuray Zengin
0104-80-26 Egemen Koşar (r)
0105-75-26 Barış Koşar (r)
0106-71-03 Erkan Naycı
0107-73-19 Recep Kaan
0108-77-09 M.Kerem Karaağaç
0109-73-34 Ali Osman Yılmaz
0110-66-34 Mustafa Gül
0111-69-34 Veli Soner
0112-76-59 Tuncay Koçoğlu
0113-72-06 Zehra Altun
0114-76-58 Ali Acungil
0115-76-20 Serkan Kılınç
0116-71-34 Ersan Altan
0117-76-60 Yılmaz Demir (r)
0118-77-59 Bora Üreden
0119-63-16 Ali Oğuz Koç (r)
0120-78-36 Cenk Başaran (r)

1.0 FARE'Yİ PROGRAMLAMAK

Bir bilgisayarınız, bir fareniz var ve siz de programcıysanız, ne yapacağınız malum! O fareyi programlarsınız! Ama nasıl? Hiç de zor değil. Farenin yapacaklarının çoğunu zaten kesintiler üstlendiğinden size o kesintileri çağırmak kalır. Peki fare kesintisinin numarası ne? fare.h dosyasındaki FARE-KESİNTİSİ sabitinin değerine bakarsanız, göreceksiniz. fare.c programındaki koddan yola çıkarak benim çevirici dilini bildiğimi sanmayın. O parçaları yazımıza çeşni katıp, bir işi birkaç yoldan yapmayı göstermek için yazdım. Eğer bir çevirici dili derleyiciniz yoksa kendiniz de programı yalnızca C derleyicisi ile derlenecek duruma dönüştürebilirsiniz. Bu yazı, Microsoft Mouse uyumlu farelerin metin ekranlarında kullanılması amacıyla yazılmıştır.

1.0 GÖRÜNTÜ SİSTEMLERİ

1.1 80x25

Bu sistem için, 640 * 200'lük mantıksal koordinatlar kullanılır. Mantıksal karakter boyu 8 * 8'dir. Diğer bir deyişle tüm yatay koordinatlar 8'in katlarıdır. Aynı zamanda tüm düşey koordinatlar 8'in katlarıdır. Ekranın en üst sol köşesinin koordinatları (0,0) ve en alt sağ köşenin koordinatları (79,24)'tür.

FARE.H

```
#include <dos.h>

#pragma inline

#define FARE_KESINTISI 0x33

/* ********************************************************** Islev No. */

int   Fare_Yerles(int *tus_sayisi);                    /* 0 */
void  Imleci_Goster(void);                             /* 1 */
void  Imleci_Sakla(void);                              /* 2 */
void  Fare_Nerede(int *tus_durumu,int *FareX, int *FareY);     /* 3 */
void  Fareyi_Gonder(int YeniX, int YeniY);             /* 4 */
void  Farenin_Yatay_Sinirlari(int sol,int sag);        /* 7 */
void  Farenin_Dusey_Sinirlari(int ust,int alt);        /* 8 */

typedef union FareTuslari {
  unsigned int i;
  struct {
    sol   :   1;
    sag   :   1;
    orta  :   1;
  } b;
} FareTuslari;

#define FareVar()  peekb(0x0000,0x00cc)
```

FARE.C

```
#include "fare.h"

int   Fare_Yerles(int *tus_sayisi){
  union REGS regs;
  if(FareVar()) {
    regs.x.ax=0;              /* Servis No. belirle     */
    int86(FARE_KESINTISI,&regs,&regs); /* Fare kesintisini cagir  */
    *tus_sayisi=regs.x.bx;    /* Tus sayisini dondur    */
    return regs.x.ax;         /* Fare durumunu dondur   */
  }
  else {
    return 0;
  }
}

void  Imleci_Goster(void) {
  asm mov ax,1;  /* _AX = 1                */
  asm int 33h;   /* geninterrupt(FARE_KESINTISI)  */
}

void Imleci_Sakla(void) {
  union REGS regs;              /*          */
  regs.x.ax=2;                  /* asm mov ax,2;  */
  int86(FARE_KESINTISI,&regs,&regs); /* asm int 33h;   */
}

void  Fare_Nerede(int *tus_durumu,int *FareX, int *FareY) {
  union REGS in,out;
  if(peekb(0x0000,0x00cc)) {
    in.x.ax=3;
    int86(FARE_KESINTISI,&in,&out);
    *FareX=out.x.cx;
    *FareY=out.x.dx;
    *tus_durumu=out.x.bx;
  }
}

void  Fareyi_Gonder(int YeniX, int YeniY) {
                /* union REGS regs;              */
  _AX=4;        /* regs.x.ax=4;                  */
  _CX=YeniX;    /* regs.x.cx=YeniX;              */
  _DX=YeniY;    /* regs.x.dx=YeniY;              */
  geninterrupt(FARE_KESINTISI);  /* int86(FARE_KESINTISI,&regs,&regs);  */
}

void  Farenin_Yatay_Sinirlari(int sol,int sag) {
  asm mov ax,7;
  asm mov cx,sol;
  asm mov dx,sag;
  asm int 33h;
}

void  Farenin_Dusey_Sinirlari(int ust,int alt) {
  asm mov ax,8;
  asm mov cx,ust;
  asm mov dx,alt;
  asm int 33h;
}
```

FARE_OR.C

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include "fare.h"
#define ESC 27

int FareBayragi, Fx, Fy, Tus_Sayisi;
FareTuslari Tus;
char c=0;
int sol, sag;
```

Not: Yine de programlarda hiç 8'le çarpma/bölme olmayışı sanırım ilginizi çekecek, çarpma/bölme yerine sola/sağa bit kaydırma işlemlerini kullandım. Bunun nedeni programı daha hızlı ve kısa bir hale getirmekti. Bu işlemler çarpma/bölmeye göre on kat hızlıdır. Ne var ki, yalnızca tamsayılarla kullanılabilirler. Bir sayıyı, bir bit sola kaydırmak (C için <<) demek, o sayıyı 2 ile çarpmak; bir bit sağa kaydırmaksa (C için >>) o sayıyı 2 ile bölmek demektir. Örneğin, bir sayıyı 8 ile çarpacağımıza (2^3=8)den sayımızı 3 bit sola kaydırırız. Başka bir soruysa şöyle olabilir: Bir sayının 16'ya bölümü kaçtır? 2^4=16 olduğuna göre, bölünen sayımızı bölmek yerine 4 bit sağa kaydırırsak, yanıtı buluruz.

1.2 40x25

Bu sistem için, 640 * 200'lük mantıksal koordinatlar kullanılır. Mantıksal karakter boyu 16 * 8'dir. Diğer bir deyişle tüm yatay koordinatlar 16'nın katlarıdır. Aynı zamanda tüm düşey koordinatlar, 8'in katlarıdır. Ekranın en üst sol köşesinin koordinatları (0,0) ve en alt sağ köşesinin koordinatları (39,24)'tür.

2.0 FARENİN TUŞLARI

Fare işlev çağrıları, fare tuşlarının şimdiki durumlarını ve eğer bir tuş belirtilirse, o tuşa kaç kez basılıp/bırakıldığı bilgisini geri döndürebilirler. Tuşların durumu bir tamsayının ilk 3 bitidir.

Bit 0: Sol tuşun durumu (0-kalkık, 1-basılı)

Bit 1: Sağ tuşun durumu (0-kalkık, 1-basılı)

Bit 2: Orta tuşun durumu (0-kalkık, 1-basılı)

Bu bilgilerden yola çıkarak fare. H dosyasında bulunan fare tuşları tipini yazdım. Fare tuşlarının herhangi bir bitine Fare Tuslari.b.sol/sag/orta yazarak ulaşılabilir.

3.0 FARE İŞLEV ÇAĞRILARI

Fare Yerleşimi
int Fare_Yerles (int*tus_sayisi)

Bu işlev, farenin takılı olup olmadığını ya da fare sürücülerinin yüklü olup olmadığını denetler. Eğer sonuç başarılıysa -1 döndürür.

Girdi
AX = 0
Çıktı
AX = fare var mı (= -1), yok mu?
BX = tuş sayısı

3.1 İmleci Göster
void Imleci_Goster(void)

Bu işlev, farenin şimdiki konumunu ekranda göstermeye yarar.

Girdi
AX = 1
Çıktı
Yok

3.2 İmleci Sakla
void Imleci_Sakla(void)

Bu işlev farenin ekrandaki görüntüsünü yok eder.

```
int ust, alt;
int ac_kapa=-1;
void main(void) {

FareBayragi=Fare_Yerles(&Tus_Sayisi);
if(FareBayragi == -1) {
  printf("Farenizin yeteneklerinin yalnizca bir kismini goreceginizi unutmayin.\n");
}
else {
  printf("Fareniz ya takili degil, ya da fare sürücünüz degil \n");
  printf("Farenizin takili sürücünüzün de yüklü oldugundan emin olunca\n");
  printf("programi çalistirmayi bir daha deneyin\n");
  exit(EXIT_FAILURE);
}

Fareyi_Gonder(79,24);
clrscr();

printf("Fareniz %s Mouse adli kardesi ile uyumlu \n",
  ((Tus_Sayisi == 3) ? "Mouse Systems" : "Microsoft"));
printf("Bastiginiz tusu ekranda görebilirsiniz \n");

Tus.i = 0;
while(Tus.i == 0)
  Fare_Nerede((int *)&Tus,&Fx,&Fy);

Imleci_Goster();
printf("Eger sol tugu tiklarsaniz, x eksenindeki sinirlarinizi daralacak \n");
printf("Eger sa  tugu tiklarsaniz, y eksenindeki sinirlarinizi daralacak \n");
printf("Eger sa +sol  tiklarsaniz, tüm ekranda gezinebilecksiniz.\n");
printf("Bu bileğimi bir daha tiklarsaniz sinirli bölgeye geri dönersiniz.\n");
printf("Programdan çikmak için ESC tusuna basin \n");

while(c!=ESC){
  Fare_Nerede((int *)&Tus,&Fx,&Fy);
  gotoxy(1,25);
  printf("X: %3d Y: %3d Tus Durumu: %3d",Fx>>3,Fy>>3,Tus.i);

if(Tus.i && 3) {     /* if(Tus.b.sol && Tus.b.sag) { */
  if(ac_kapa<0) {
    Farenin_Yatay_Sinirlari(0<<3,79<<3);
    Farenin_Dusey_Sinirlari(0<<3,24<<3);
    ac_kapa*=-1;
  } else {
    Farenin_Yatay_Sinirlari(sol<<3,sag<<3);
    Farenin_Dusey_Sinirlari(ust<<3,alt<<3);
  }
  ac_kapa*=-1;
}

  gotoxy(32,25);
  if(Tus.b.sol && !Tus.b.sag) {
    printf("SOL");
    if(sol<sag) {
      ++sol;

      --sag;
    } else {
      sol=0;
      sag=79;
    }
    Farenin_Yatay_Sinirlari(sol<<3,sag<<3);
  }
  else {
    printf("   ");
  }

  gotoxy(36,25);
  if(Tus.b.orta) {
    printf("ORTA");
  }
  else {
    printf("    ");
  }


  gotoxy(41,25);
  if(Tus.b.sag && !Tus.b.sol) {
    printf("SAG");
    if(ust<alt) {
      ++ust;
      --alt;
    } else {
      ust=0;
      alt=24;
    }
    Farenin_Dusey_Sinirlari(ust<<3,alt<<3);
  }
  else {
      printf("   ");
    }
    if(kbhit())
      c=getch();

  }

  Imleci_Sakla();
}
```

Girdi
AX = 2
Çıktı
Yok

3.3 Fare Nerede
void Fare_Nerede(int*tus_durumu,int*FareX,int*FareY)

Bu işlev, farenin şimdiki konumunu ve tuş durumunu bildirir. Anımsarsanız basılı tuşun bit değeri, 1, basılı olamayan tuşun değeri 0'dır.

Girdi
AX = 3
Çıktı
BX = tuş durumu /*tus_durumu*/
CX = yatay konum/*FareX */
DX = düşey konum/* FareY */

3.4 Fareyi Gönder
void Fareyi_Gonder(int YeniX, int YeniY)

Bu işlev, fare imlecini ekranda istediğiniz yere gönderir.

Girdi
AX = 4
CX = Yeni yatay konum/*YeniX*/
DX = Yeni dikey konum/*YeniY*/
Çıktı
Yok

3.5 Bu işlevleri şimdilik boş bırakıyorum. Bu nedenle
3.6 Bu numaralara herhangi bir işlev koymadım.
3.7 Farenin Yatay Sınırları
void Farenin_Yatay_Sinirlari(int sol,int sag)

Bu işlev, farenin yatay hareket alanını belirler.

Girdi
AX = 7
CX = en küçük X değeri/*sol*/
DX = en büyük X de eri/*sag*/
Çıktı
Yok

3.8 Farenin Düşey Sınırları
void Farenin_Dusey_Sinirlari(ins ust, int alt)

Bu işlev, farenin düşey hareket alanını belirler.

Girdi
AX = 8
CX = en küçük Y değeri/*ust*/
DX = en büyük Y de eri/*alt*/
Çıktı
Yok

4.0 SONUÇ

Bu programı Turbo C v2.0 ve Turbo Assembler v2.0 ile derledim. Denemelerimi de Microsoft Mouse ve TRUEDOX Mouse sürücüleri ile yaptım. fare.c programını yalnızca derleyip, fare.obj dosyanın oluşmasını sağlayın. fare_or.c programını da fare.obj ile birlikte derlemeyi unutmayın.

Turbo C için:
tcc -c fare.c
tcc fare_or.c fare.obj

066-71-34 Mahmut Emin Yazıcı Beyçayırı Sok. Kale Apt. 43/1 D: 4, 34290 Kocamustafapaşa, İstanbul

Halen İstanbul Üniversitesi Biyoloji Bölümü'nde okumakta olan üyemiz, Basic ve Pascal biliyor, çeşitli veritabanı, elektronik tablolama ve kelime işlemci programları kullanıyor. Kendisine ait 386 işlemcili compatible Escort bilgisayarı var.